Improving Performance of the Convolutional Neural Networks for Electricity Theft Detection by using Cheetah Optimization Algorithm

H. Ghaedi[1], S. R. Kamel Tabbakh[2*], R. Ghaemi[3]
1-Department of Computer, Neyshabur Branch, Islamic Azad University, Neyshabur, Iran.
Email: H.ghaedi@iau-neyshabur.ac.ir
2-Department of Computer, Mashhad Branch, Islamic Azad University, Mashhad, Iran.
Email: rezakamel@ieee.org(Corresponding author)
3-Department of Computer, Quchan Branch, Islamic Azad University, Quchan, Iran.
Email: rezaghaemi73@gmail.com

Received: 4 April 2022 Revised: 12 May 2022 Accepted: 30 June 2022

ABSTRACT:
Today, electricity theft is one of the main challenges for energy distribution and transmission companies around the world. Early detection of abnormal consumers can prevent security and financial losses. Extensive research studies have been done to detect electricity theft by analyzing customer consumption patterns. Today, one of the most widely used methods is convolutional neural networks (CNNs). These networks contain a large number of hyper-parameters. The accuracy of these networks is low in most studies due to the lack of attention to the adjustment of these hyper-parameters. Network accuracy and achieving a robust learning model are influenced by the optimal adjusting of these hyper-parameters, which requires exploring a complex and large search space. Meta-heuristic-based search methods are suitable for solving these problems. Therefore, the main contribution of this paper is to use the high ability of the cheetah optimization algorithm (CHOA) to optimally extract CNN hyper-parameters. In this paper, in order to balance the dataset, abnormal samples are created using artificial attacks and added to the dataset. Also, in order to increase the accuracy of the network, abnormal data are clustered using the CHOA algorithm. ISSDA dataset is used to test and evaluate the results. Based on the results obtained and comparing them with the other works, it was proved that the proposed framework with high accuracy identifies abnormal consumers.

KEYWORDS: Data mining, Classification, Electricity Theft Detection, Convolutional Neural Network (CNN).

1. INTRODUCTION

Electricity theft is one of the major challenges for energy companies. If electricity theft is not detected in the early stages and the number of abnormal consumers are high, the network will suffer irreparable damage. Due to this, several research studies have been done to detect electricity theft using traditional data mining algorithms such as KNN, SVM, Stacking, NB, and etc.

Jokar et al.[1] presented a consumption pattern-based power theft detector that aids in identifying normal and abnormal consumption patterns of the customers. They used a combination of clustering and classification in parallel with the use of transformer meters and anomaly detectors to robust their algorithm against unauthorized changes. The values obtained for false positive rate (FPR) and detection rate (DR) criteria are 11% and 94%, respectively. Ghaedi et al.[2] improved the crow search algorithm (CSA)[3] for the optimal extraction of support vector machine. They used the weight and awareness probability (AP) of crows to improve operations. In order to balance the dataset used, artificial attacks have been added to the dataset. The used accuracy (0.9993), recall (0.97), precision (0.97), and f-score (0.97) criteria for evaluation. Razavi et al. [4] proposed a feature-based engineering framework for detecting power theft on smart grids. The proposed framework is a combination of the finite mixture model (FMM) clustering for customer segmentation and a genetic algorithm for identifying new features. Due to the large volume of demand (4000 home customers), the gradient Boosting machine (GBM) algorithm has been used and the area under the ROC Curve (AUC) obtained was 0.811.

Feng et al.[5] developed an algorithm based on local matrix reconstruction (LMR). They were used 5 daily load characteristics to remove high dimensional daily load curves. Moreover, principal component analysis (PCA) was used to calculate the weighted reconstruction error in a local area. The amount of outlier was calculated by comparing the reconstruction error of each sample to that of adjacent

Paper type: Research paper
DOI: 10.30486/mjee.2022.696521
How to cite this paper: H. Ghaedi, S. R. Kamel Tabbakh and R. Ghaemi, “Improving Performance of the Convolutional Neural Networks for Electricity Theft Detection by using Cheetah Optimization Algorithm”, Majlesi Journal of Electrical Engineering, Vol. 16, No. 4, pp. 103-115, 2022.

samples, which indicates the abnormal degree of each load sample. In this study, the AUC for large datasets was 0.8463. In an innovative study, Ghaedi et al. [6] proposed a new algorithm so-called the cheetah optimization algorithm (CHOA) to increase the correct rate of electricity theft detection, and select important features in the stacking technique. They used the Naïve Bayes (NB), support vector machine (SVM), AdaBoost selected algorithms as the base classifiers, and the single-layer perceptron (SLP) model to obtain the final result. They used the accuracy (0.996), recall (0.9984), precision (0.9969), and f-score (0.9977) criteria for evaluation.

Low accuracy, use of one-dimensional electricity consumption data, failure to record the periodicity of customers' electricity consumption, an imbalance between normal and abnormal samples, and lack of short-term time data are the challenges of these research studies. As a result , the CNN network is a type of deep learning model that is widely used today to detect electricity theft. This type of network is composed of different layers of convolution with numerous hyper-parameters. The research studies that have been conducted to detect electricity theft using CNN networks are summarized here.

Ullah et al.[7] combined CNN network with particle swarm optimization (PSO) and gated recurrent unit (GRU) to detect power theft, overcome the challenge of over-fitting, and manage unbalanced data. The data was collected from antenna-based smart meters installed at the end of the consumers. Firstly, they processed the data to select and extract the optimal features using the local mean method. Then they have done feature engineering with CNN. They used the accuracy (0.94), AUC (0.95), f-score (0.94), and precision (0.94) metrics to evaluate performance. Ibrahim et al.[8] developed a CNN model for the automatic detection of power theft. They performed several experiments to find the best sequential model configurationfor classifying and detecting power theft. The best performance was obtained from a two-layer model with the first layer having 128 neurons and the second layer having 64 neurons. They also used the blue monkey (BM) algorithm to reduce the number of data features. The prediction accuracy was 0.92. Paper[9] presented a power theft detection system based on a combination of a CNN model and the LSTM architecture. Because the effect of power consumption is time series, a combined CNN-LSTM model is used to classify smart grid data. In the dataset used, the number of electricity theft customers is relatively small and for this purpose, the synthetic minority over-sampling technique (SMOTE) has been used to produce artificial examples of theft. They used accuracy(0.89), precision (0.90), and recall (0.87) metrics to evaluate the results. Li et al. [10] have developed a CNN-random forest (RF) hybrid model for automatic detection of power theft to help power companies in solving problems. In this model, a CNN network is first designed to learn features. A dropout layer has also been added to delay the risk of over-fitting. The back-propagation algorithm is also used to update network parameters in the training phase. The random forest is then trained to determine whether the consumer is a thief or not. To construct a random forest, a grid search was used to determine the optimal parameters and SMOTE technique was used to solve the unbalanced data problem. They used the AUC (0.98), recall (0.97), precision (0.97), and f-score (0.97) criteria for evaluation. Kocaman and Tumen [11] presented an efficient method for detecting power theft using deep learning methods based on actual daily power consumption data. Data reduction is used to make the dataset more efficient and extract more meaningful results. Because the dataset contains time series data, the deep learning-based LSTM model is used to extract high-level features. They used accuracy (0.936), precision (0.9165), and recall (0.906) criteria to evaluate the results. Rouzbahani et al. [12] developed a hybrid deep CNN to detect power theft. In the first layer of this model, a random bagging technique is applied to unbalanced data. CNN networks are then used in each subset. Finally, a voting system is used in the last layer. Evaluation of results is based on AUC (0.993), precision (0.988), and recall (0.99). Zheng et al. [13] developed a method of detecting power theft based on Wide & Deep CNN to solve problems and challenges in power generation networks. Based on 2D power consumption data, they used the ability of the CNN Deep component to detect non-periodic power thefts and periodic normal consumption. The ability of the Wide component to capture the general characteristics of one-dimensional power consumption data has also been used. They achieved an AUC of 0.768 and 0.7815 with 70% and 80% of the training data, respectively. Later, Chandel and Thakur [14] combined CNN with a two-sided LSTM-based recurrent neural network (RRN) network to overcome the problem of one-dimensional power consumption data to detect power theft. CNN records one-dimensional general variables and identifies two-dimensional records of periodic and non-periodic power consumption. Using BiLSTM-based RRN, they have expanded the neural network memory capacity with two- sided information flow. The AUC obtained from the implementation of the presented method was 0.961. Feng et al. [15] presented a scheme for detecting power theft based on textCNN networks. They have converted electricity consumption measurements into 2D time series that represent the characteristics of electricity consumption around the clock. They have also proposed a data enhancement method to deal with power theft data imbalances. The values of the accuracy, precision, recall, and f-score for 80% of the training data of the dataset ISSDA are 0.958, 0.857, 1, and 0.947, respectively. Maamar and Benahmed [16] presented a combined method based on a combination of K-means and deep neural network (DNN) to detect power theft anomalies in AMI systems. K-means is used to identify different types of normal behaviors. DNN is used to construct an anomaly detection model that can detect behavioral changes or anomalies. They used FPR (8.86%) and DR (95.38%) criteria for evaluation.

There are several challenges in designing the CNN architecture, including high computational costs for information processing and determining the optimal hyper-parameters for each problem. CNN architecture is made up of various parameters that can generate different classification results for a particular problem depending on their configuration. Adjusting hyper-parameters is one of the most important challenges of this type of network. Optimal adjustment of these hyper-parameters has a direct impact on network performance and achieves a robust learning model that requires exploring in a large and complex search space. Manually adjusting them is also boring and time-consuming and sometimes impossible. Several studies in the field of

adjustng the hyper-parameters of CNN networks using meta-heuristic algorithms are discussed below.

Using the PSO algorithm, Fouad et al.[17] proposed a design architecture for the CNN network to learn the optimal values of its hyper-parameters. They optimized the convolution kernel size, the number of kernels, and neurons numbers in the fully connected layer and used the MNIST dataset and achieved test error 44%. Bacanin et al. [18] used advanced versions of tree growth algorithm (TGA) and firefly algorithm (FA) algorithms to optimize CNN hyper-parameters in their study. They improved the design of the CNN network on the MNIST dataset and optimized the number of convolutional layers, the number of fully connected layers, the number of kernels per convolutional layer, kernel sizes, and size of each fully connected layer. Brodzicki et al. [19] used the Whale optimization algorithm (WOA) to optimize CNN hyper-parameters. They compared the proposed algorithm with the well-known grid and random search algorithms. They used a combination of epochs-batch size and epochs-optimizer to optimize the network. The simulations showed that the proposed algorithm for the Fashion MNIST and Reuters datasets achieved the accuracy values of 89.85% and 80.60%, respectively. Han et al. [20] used the LSTM network to predict short-term loads and the PSO algorithm to optimize network parameters. They used meteorological data and historical load data of a certain place as the input of the LSTM network and compared the proposed model with back propagation (BP) neural network. They optimized the number of neuron nodes, learning rate and the number of iterations in the hidden layer of the LSTM. The results showed that the PSO-LSTM model had higher reliability and predictive accuracy. Kim et al.[21] used harmony search algorithm (HS) to optimize the hyper-parameters of CNN network in the human respiration pattern recognition system. The paraneters of kernel size for convolution layer, kernel count for convolution layer, and dense layer (FC Layer) neuron count were optimized. By optimizing the hyper-parameters, they achieved an efficiency of 96.7%. Serizawa & Fujita [22] used the linearly decreasing weight pso (LDWPSO) algorithm to extract the optimal hyper-parameters of the CNN network. They optimized the number of filters and the size of filters in convolutional layers, activation function in convolutional and fully connected layers, the number of neurons in fully connected layers, Batch size, and optimizer. They achieved accuracy values of 98.95% and 69.33% for the MNIST and cifar- 10 datasets, respectively, which are more efficient than the CNN baseline. Li et al.[23] used genetic algorithm to optimize the hyper-parameters of CNN network. The proposed system in the Cats and Dogs datasets has converged to 97% accuracy for the best state. Yeh et al.[24] used a simplified swarm optimization (SSO)-based algorithm to optimize the network hyper-parameters. They optimized the original LeNet model such as the number of kernels per convolution layer, the stride of convolution layers, the size of pooling layers, etc. The results show that the accuracy values of the SSO-CNN method for MNIST, Fashion MNIST, and cifar-10 datasets are 99.58%, 92.75%, and 73.13%, respectively. Zhou [25] used a genetic algorithm to optimize the CNN network for chest X-ray classification. In their works, the dimensions and activation functions of convolutional layers, the number of fully connected layers and drop rate were optimized. Compared to the pre-trained VGG 16 network[26], the presented framework performs the classification with 74.9% accuracy. Kan et al.[27] used PSO algorithm to optimize the hyper-parameters of filter numbers, filter sizes, activation function, Batch size and, learning rate of the CNN light weight network for the interpretation of motion information of surface electromyography (sEMG). Three experiments were performed on Ninapro dataset 1(NB1) and achieved the accuracy values of 86.67%, 90.06%, and 89.57%. Johnson et al.[28] used a genetic algorithm to optimize the kernel sizes and the number of filters per layer, as well as the number of CNN network. The accuracy values obtained for cifar-10, MNIST and Caltech256 datasets are 84.85%, 99.56% and .333%, respectively, which is more efficient than the compared works. Aszemi & Dominic [29] used a genetic algorithm to adjust the number of filters, kernel size, and activation function of the CNN network on the cifar- 10 dataset.

In research studies that were analyzed in the field of electricity theft detection, items such as the number of convolution and pooling layers, the number of filters per convolution layer and their size, stride per filter, pool size and stride per pooling layer and batch size were not optimally adjusted, but rather manually adjusted. The main contribution is to provide an efficient framework for extracting optimal CNN network hyper-parameters using innovative CHOA algorithm. The CHOA algorithm is also used to cluster abnormal samples created by artificial attacks. Abnormal data clustering before classification increases the classification accuracy rate.

In general, this paper is organized as follows: Section 2 reviews the convolutional neural network; Section 3 describes the CHOA algorithm. Proposed framework is explained in Section 4; Experiments and evaluations are discussed in Section 5 and Section 6 is dedicated to the paper's conclusions.

2. CONVOLUTIONAL NEURAL NETWORKS (CNN)

CNN is a model of multilayer neural network that typically includes convolutional, pooling, and fully connected layers [30]. The input of CNN is a multilayer or deep matrix such as an image and the layers of this network consist of three-dimensional neurons, which in some studies are called tensors. In these networks, single-layer neurons are connected to only a small area of the previous layer and not to all previous layer neurons

.

2.1. Convolutional Layer

This layer is considered the main layer in CNNs. The principal duty of this layer is to extract features. This layer applies convolutional operations to the input data. The outputs are obtained from this layer and called feature maps. As a result, all neurons in a feature map have a set of similar and common weights and biases that make the image features recognizable in different positions. In CNNs, connections are made in small and local areas. In other words, each neuron in the first hidden layer is connected to a small area of the input neurons. For example, this area can be 2× 2. This small 4-pixel area is called filter or convolutional kernel [31]. For further explanation, Fig 1 shows an input image 4×4. The kernel window 2×2 moves from left to right on the input

pixels. Each window is connected to a neuron in the hidden layer. Thus, the hidden layer will include a 3× 3 neural network, as shown in Fig 1,.

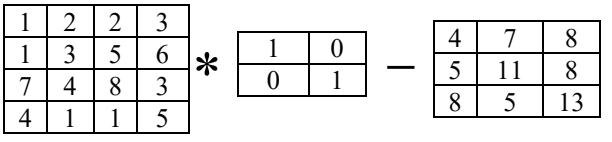

Fig .1. Using kernel in the convolutional layer

Applying the activation function to the convolutional layer increases the nonlinear properties of the output. There are different types of activation functions. Some examples of the most popular activation functions are the sigmoid function, the hyperbolic tangent function (tanh), the linear rectifier function (ReLU), the exponential linear function (ELU), and the Softmax. The number of parameters that the network must be trained in a convolutional layer is very large.

2.2. Pooling Layer

In a convolutional neural network, a pooling layer is usually placed after each layer of convolutional. This layer is important because it reduces the number of parameters that need to be trained [32]. Therefore, while this layer reduces the number of necessary calculations in the training section, it also controls possible over-fitting in the network.This layer is applied to each cut of the deep input layers and resizes them. Maxpooling and Average pooling are the two most popular functions of this layer, with the first model being more often used on CNNs. Maxpooling works by sending the largest pixel to the output in each window. This window moves on the input as a convolutional function from left to right and from top to bottom with the specified stride size and sends the result to the output. The pooling operation is usually done with a 2×2 filter. Suppose there is a 4×4 feature map obtained from the convolutional layer. Fig 2 shows how to apply Maxpooling with stride 2.

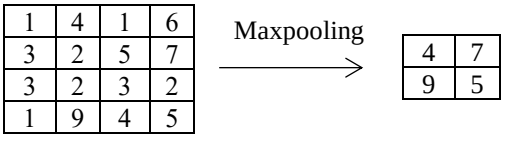

Fig .2. Using Maxpooling in the pooling layer

The output depth is the same as the input depth to the pooling layer and does not change because this operation is applied to all cuts,

2.3. Fully Connected Layer

The fully connected layer is the last layer of a convolutional neural network and makes perfect connections to the output of the previous layer. This layer receives the input and then generates the output as a vector with N components [33]. The number of classes that the network classifies are equal to N. In learning algorithms, the important issue is to consider the relationship between the data class and the assumption obtained from the result of the algorithm training. This relationship, also known as the loss or cost function, must be optimized by appropriate methods. Important optimization methods for a CNN include descending gradient, descending batch gradient, random descending gradient and mini-batch descending gradient, Adagrad algorithm, RMSprop algorithm, and Adam. In CNNs, the network is prone to over-fitting. Many different ways have been suggested to solve this problem. One popular method is to increase the number of training data and the other method is to add a suitable dropout layer to the fully connected layer to solve this problem. The method of increasing data in the subject of learning processes is a common method of increasing the number of images entering the network by applying changes in images such as rotation, magnification, moving, etc. In the dropout method, before each training step, a number of neurons are randomly removed from the training process in the proportion specified in the program, and only the remaining neurons participate in the training [34]. In the fully connected layer, the Softmax function is always used for classification tasks [35].

3. CHOA ALGORITHM

The innovative CHOA algorithm[6] is a population-based meta-heuristic algorithm in which the location and velocity of the cheetahs or victims are specified in the search space. All cheetahs choose a direction according to the movement of the victim, and the victim chooses its direction according to $\text{cheetah}_{\text{leader}}$. Based on the current velocity of a cheetah and its distance from the victim, a new velocity is calculated for a cheetah. To find the best solution by $\text{cheetah}_{\text{leader}}$, firstly, all of the N cheetahs and M victims are initialized at random locations in the search space. In the beginning, the velocity of each cheetah and victim is considered zero. In the next steps, the velocities and locations are updated. For each cheetah, the fitness of the new location is evaluated, and depending on the obtained value, the $\text{cheetah}_{\text{leader}}$ may be updated. If the $\text{cheetah}_{\text{leader}}$ improves, the victim will escape and its velocity and location will be updated. One of the important criteria for starting an attack is Attention. This factor is calculated for each cheetah and victim. To attack, the $\text{cheetah}_{\text{leader}}$ must have more Attention than victim. The cheetah with the highest fitness is known as $\text{cheetah}_{\text{leader}}$. The movement of each cheetah is affected by the movement of the victim. During the hunting process, if $\text{cheetah}_{\text{leader}}$ improves his fitness, it will not be able to catch the victim and the new location of the victim will be updated.

4. PROPOSED FRAMEWOKR

This section proposes an optimization framework that uses the CHOA algorithm to optimize CNN network hyper-parameters, this approach is called CHOA-CNN. The main contribution is to select the most effective hyper-parameters that affect the performance of CNN network which uses the CHOA algorithm to find these optimal hyper-parameters.

The hyper-parameters used for optimization were selected after conducting research studies on the optimization of hyper-parameters of this type of network using metaheuristic algorithms.

The different values of these hyper-parameters create a variety of architectures for CNN. For this reason, the main contribution is to find the optimal hyper-parameters. The hyper-parameters selected below were used to optimize the CNN network.

- The number of convolution and pooling layers
- The number of filters per convolution layer and their

size
- Stride per filter
- Pool size and stride per pooling layer
- Batch size

The proposed general framework is shown in Fig 3. As can be seen, in the input data block, the data are preprocessed and abnormal samples are generated and clustered.

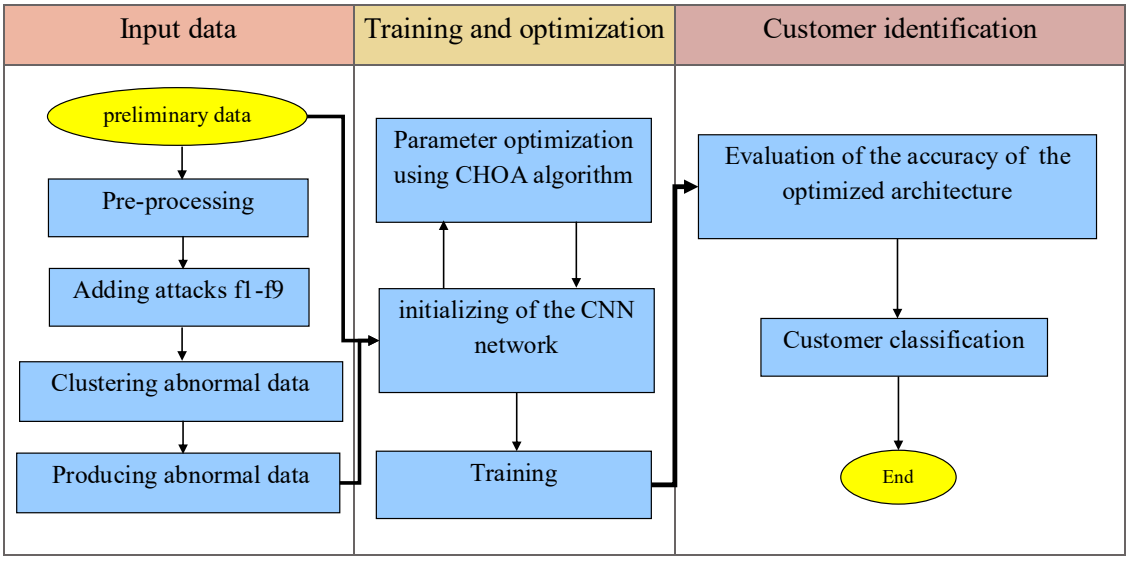

Fig .3. Steps of proposed framework

.

In the training and optimization block, the CNN network optimization process is performed using the CHOA algorithm. In this process, the CHOA algorithm is initialized based on the execution parameters and $\text{cheetah}_{\text{leader}}$ is generated. $\text{cheetah}_{\text{leader}}$ is an optimal solution that contains the optimal hyper-parameters of the CNN network. In the customer identification block, the accuracy of the optimized CNN architecture is evaluated and the label of the new sample is specified. In the following subsections, all the steps of the proposed framework are fully described.

4.1. CHOA for Clustering Abnormal Samples

This section describes the application of the CHOA algorithm to cluster abnormal electricity customers. Given that the ISSDA[36] dataset contains only normal samples of customer consumption, abnormal samples are created by artificial attacks to increase the rate of correct theft detection, as presented in Table 1.

Table 1. Artificial attacks to generate abnormal samples

Attack	Formula
$f_1(y_t)$	$\alpha \times y_t$ α=random(0.1,0.8)[1]
$f_2(y_t)$	$\alpha_t \times y_t$ α_t=random(0.1,0.8)[1]
$f_3(y_t)$	$\begin{cases} 0 & \forall t \in [ts, tf] \\ y_t & \text{otherwise} \end{cases}$ [1]
$f_4(y_t)$	$a_t \times \text{mean}(y)$ a_t=random(0.1,0.8)[1]
$f_5(y_t)$	mean(y) [1]
$f_6(y_t)$	y_{24-t} [1]
$f_7(y_t)$	min(y)
$f_8(y_t)$	$\alpha \times \text{mean}(y)$ α=random(0.1,0.8)
$f_9(y_t)$	$y_t - \min(y)$

According to Table 6, for attacks $f_1(y_t)$ - $f_6(y_t)$, Jokar research[1] was used. Also, we added attacks $f_7(y_t)$ - $f_9(y_t)$ to the dataset ourselves. $f_1(y_t)$ multiplies all features in a constant random value. $f_2(y_t)$ multiplies each feature in a random value. $f_3(y_t)$ is a by-pass attack that sends a value of zero at a specified time interval [ts, tf], otherwise, it sends the actual amount of consumption. ts and tf are the beginning and end of the theft period, respectively. $f_4(y_t)$ multiplies average readings in a random value. $f_5(y_t)$ is average readings during a day and $f_6(y_t)$ reverses the order of the readings in a day.

$f_7(y_t)$ sends the lowest features to the center. $f_8(y_t)$ multiplies the average readings in a constant random value. $f_9(y_t)$ subtracts the least amount of features from all readings.

For automatic clustering, CHOA algorithm and paper technique[37] were used. In this technique, a factor called activation threshold was used for each cluster center, and if its value is more than 0.5, the corresponding cluster center is activated. The Davies and Bouldin (DB) index[38] was used to evaluate the clustering of the CHOA algorithm. After running the clustering algorithm, the number of optimal clusters is determined and the cluster number of each abnormal sample onsidered as the class label of that data. In this study, the abnormal data were divided into 5 and 7 clusters after using the CHOA algorithm for clustering.

4.2. Optimization of the CNN Network by using CHOA Algorithm

This section presents an optimization approach that uses the CHOA algorithm to optimize the parameters of the proposed CNN architecture, named CHOA-CNN.

Reasons for using the CHOA algorithm:

1- Using the Attention factor for cheetahs and victims, which is used to select the $\text{cheetah}_{\text{leader}}$ and victim
2- Two-step update of velocity and location of cheetahs and victim.
3- Rapid convergence of the CNN network to the optimal solution
4- Do not trap the algorithm in local optimization
5- Good balance between diversification and intensification[6]

Firstly, the CHOA algorithm is initialized based on the execution parameters and $\text{cheetah}_{\text{leader}}$ selected. $\text{cheetah}_{\text{leader}}$ is the possible solution to the problem. Its location contains optimized hyper-parameters. The training step is repeated until all cheetahs produced by the CHOA algorithm were evaluated in each iteration. Fig 4 shows the steps of optimizing the CNN architecture by the CHOA algorithm. The parameters of the CHOA algorithm were adjusted after loading the data, as shown in Fig 4.

Table 2. Static parameters for CNN architecture and CHOA algorithm

Parameter	Value
Learning function	Adam
activation function(classifying layer)	Softmax
non-linearity activation function	ReLU
Victims	10
Cheetahs	10
Iteration	100
Rhunting	10

Table 3. Search spaces of cheetahs and victims

Hyper parameter	Description	Search space
L1	Number of convolutional and pooling layers	[1,3]
L2	Filter numbers of the first convolutional layer	[8,32]
L3	Convolutional filter size of the first layer	[1,3]
L4	Strides of the first convolutional layer	[1,4]
L5	Pool size of the first Maxpooling layer	[1,3]
L6	Strides of the first Maxpooling layer	[1,4]
L7	Filter numbers of the second convolutional layer	[16,64]
L8	Convolutional filter size of the second layer	[1,3]
L9	Strides of the second convolutional layer	[1,4]
L10	Pool size of the second Maxpooling layer	[1,3]
L11	Strides of the second Maxpooling layer	[1,4]
L12	Filter numbers of the third convolutional layer	[32,128]
L13	Convolutional filter size of the third layer	[1,3]
L14	Strides of the third convolutional layer	[1,4]
L15	Pool size of third Maxpooling layer	[1,3]
L16	Strides of the third Maxpooling layer	[1,4]
L17	Batch size	[8,128]

In the presented CHOA-CNN framework, the cheetah and victim architecture consists of 17 locations, as shown in Table 3. Every location is a hyper-parameter that must be optimized.

These parameters include the number of cheetahs and victim, Rhunting (The radius of the cheetah group), the number of iterations, learning function, activation function (classifying layer), non-linearity activation function, as presented in Table 2. This step also includes the designing of cheetahs and victims, the architecture of which is shown inTable 3. Then, the CNN architecture is initialized, trained, validated, and tested with the hyper-parameters obtained by the CHOA algorithm. The value of the obtained fitness function is also returned to the CHOA algorithm. Attention factor is then calculated for each cheetah and victim. One of the cheetahs who has higher fitness than the others is chosen as the $\text{cheetah}_{\text{leader}}$. Then one of the victims is randomly selected as the main victim to attack. As the $\text{cheetah}_{\text{leader}}$ moves and attacks the victim, the velocity and location of the cheetahs are updated and the new location is evaluated. If the objective function is improved, the $\text{cheetah}_{\text{leader}}$ is changed and the victim's velocity and location are updated. These steps continue until the algorithm is completed. Finally, the optimal solution is selected. The optimal solution is actually the $\text{cheetah}_{\text{leader}}$, which contains the optimal hyper-parameters of CNN architecture.

Table 4. Filter sizes of convolution layer

L3, L8 and L13	Search space
1	[1,1]
2	[2,2]
3	[3,3]

Table 5. Strides of convolution and Maxpooling layers

L4, L6, L9, L11, L14 and L16	Search space
1	[1,1]
2	[2,1]
3	[2,2]
4	[3,1]

Table 6. Pool sizes of Maxpooling layer

L5, L10 and L15	Search space
1	[1,1]
2	[2,2]
3	[3,3]

Tables 4, 5, and 6 describe the filter size of the

convolutional layer, strides of the convolutional and Maxpooling layers, and pool size of the Maxpooling layer in more detail. Based on the values to be optimized for the CNN architecture, the L1 location is used to identify the number of convolutional and pooling layers and activate the L2 to L16 locations. If the CHOA algorithm produces a cheetah with a value of 1 for L1, only locations L2 to L6 are activated. In other words, if the CHOA algorithm generates a cheetah with a value of 3 in location L1, locations L2 to L16 are activated according to Table 3. These values are completely distinct from each other, and therefore, this approach helps to generate different CNN networks.

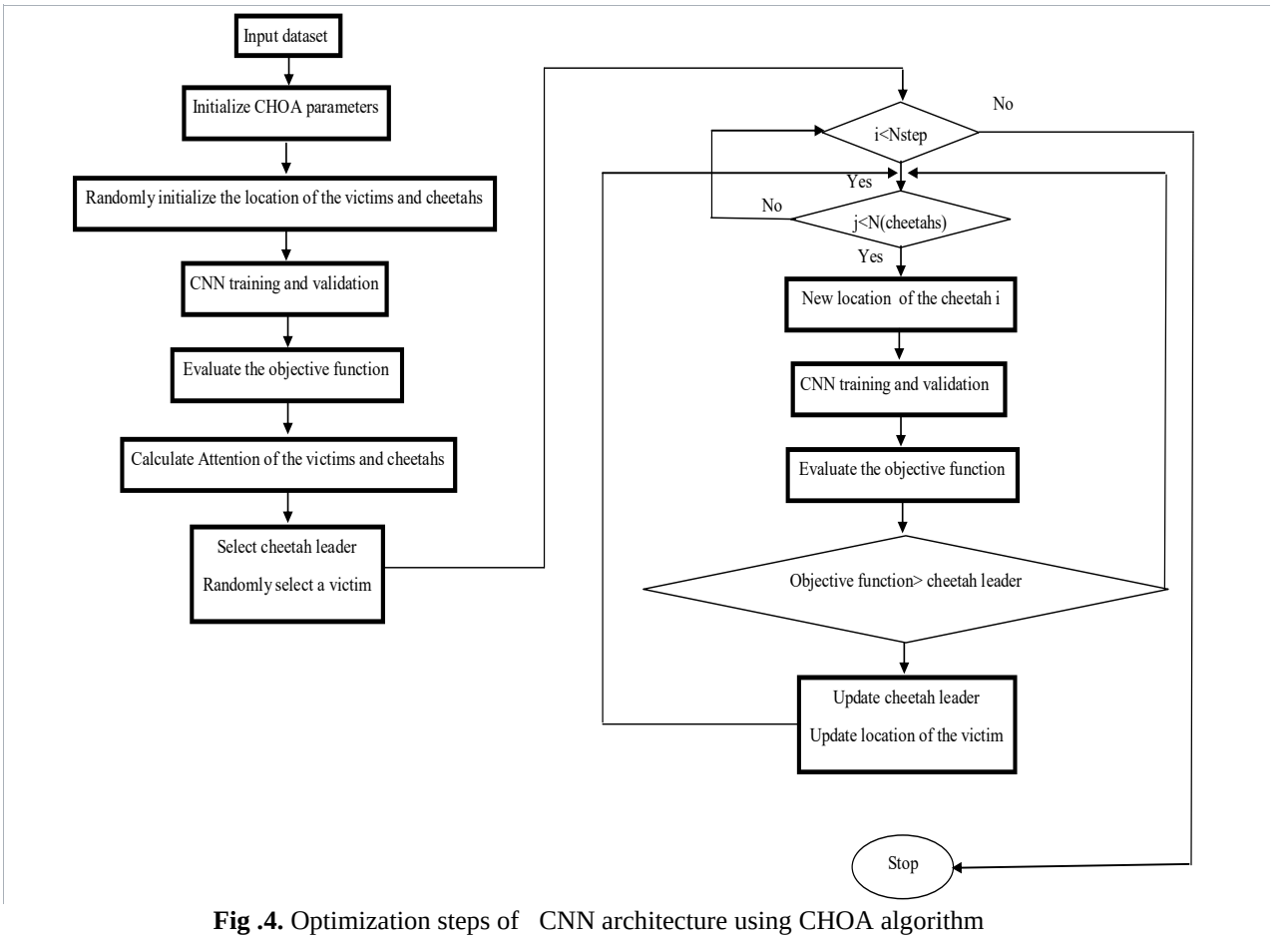

Fig .4. Optimization steps of CNN architecture using CHOA algorithm

4.3. Binary form of the CHOA-CNN architecture

Representation of the problem is one of the principal steps in adjusting CNN network hyper-parameters using the CHOA algorithm. The problem must be formulated in a method that is appropriate for the CHOA algorithm.

Table 7. Hyper-parameters in the binary form

L1=1	L1=3	L7=16	L7=64	L14=1	L14=4
01	11	0010000	1000000	001	100

Hyper-parameters that are adjusted to their optimal values, produce the optimal CNN network and prepare the highest performance in the classification. To achieve this goal, a binary form of hyper-parameters is implemented. Table 7 shows a representation of the binary code conversion process of the L1, L7, and L14 hyper-parameters for some of the values within the acceptable search range. Each hyper-parameter is represented by its own binary form. For hyper-parameter L1, the minimum value is 1 and the highest value is 3. A maximum of 2 bits are required to represent the binary form of L1, or the minimum and maximum values of the L7 hyper-parameter are 16 and 64, respectively, and a maximum of 7 bits are required to represent it.

0.26	0.29	0.31	0.76	0.22	0.15	0.87
0.67	0.53	0.4	0.12	0.11	0.7	0.54
0.02	0.19	0.53	0.92	0.49	0.67	0.16
0.08	0.99	0.37	0.42	0.03	0.12	0.51
0.59	0.09	0.65	0.72	0.82	0.11	0.45
0.39	0.18	0.55	0.64	0.19	0.66	0.04
0.77	0.37	0.19	0.22	0.67	0.99	0.65
0.78	0.11	0.08	0.04	0.91	0.18	0.56
0.03	0.19	0.37	0.89	0.45		

Fig .5. Actual values for location of a cheetah

Since the actual values generated by the CHOA algorithm are decimal, the locations are rounded to the nearest integer to obtain the binary bit string. Fig 5 shows the actual values obtained for the location of a cheetah. As you can see, all the values related to the location of this cheetah are decimal. In

order to map these values to different hyper-parameters, these values are rounded to the nearest integer as shown in Fig 6. For this study, a population of 10 cheetahs was formed. Each of these 10 CNN architectures is evaluated based on the fitness function, which is described in the next subsection.

0	0	0	1	0	0	1
1	1	0	0	0	1	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
1	0	1	1	1	0	0
0	0	1	1	0	1	0
1	0	0	0	1	1	1
1	0	0	0	1	0	1
0	0	0	1	0		

Fig .6. Rounded values related to Fig 4

5. EXPERIMENTS AND EVALUATIONS

In this section, the results obtained from various experiments of the proposed method and comparison of the results with other research studies are given.

All the experiments are conducted using Keras interface running on a desktop system with an Intel Core i11700-7K CPU and an ASUS Phoenix GeForce RTX 3060 GPU.

5.1. Dataset

The ISSDA dataset, which contains electricity consumption information for 5,000 Irish home and business customers, was used to evaluate the proposed method. This information was collected between 2009 and 2010. In this dataset, information related to 535 days of consumption of each customer is stored so that the consumption rate is sent to the center every half hour. In this study, the sampling rate is changed to each hour. Table 8 shows the details of the ISSDA dataset for a special meter from a customer. The Cons is an abbreviation for consumption.

Table 8. Details of ISSDA dataset

Meter ID	Cons in 00:00 o'clock (Kwh)	…	Cons in 23:00 o'clock (Kwh)	Cons date
1184	0.094		0.149	2009/29/8

Meter ID represents ID of the customer meter. Cons between 00:00 to 23:00 o'clock, identify consumption values in each clock. Cons date represents consumption day in a month of the year.

5.2. Evaluation Criteria

In order to evaluate the performance of the proposed framework, the evaluation criteria including accuracy, precision, recall and f-score were used and determined for two positive and negative classes according to the Table 9.

Table 9: Confusion Matrix

		Predicted Label	
		Positive	Negative
Actual Label	Positive	TP	FN
	Negative	FP	TN

TP: Data belongs to the Positive class and was correctly categorized.

FN: Data belongs to the Positive class but was incorrectly assigned to the Negative class.

TN: Data belongs to the Negative class and was correctly categorized.

FP: Data belongs to the Negative class but was incorrectly assigned to the Positive class.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F - Measure = \frac{2 * (Precision * Recall)}{Precision + Recall} \quad (4)$$

5.3. Analysis of results

To evaluate the performance of the proposed framework and the possibility of comparison with other research studies, experiments were performed in two parts. In the first part, attacks $f_1(y_t)$ to $f_6(y_t)$ and in the second part, attacks $f_1(y_t)$ to $f_9(y_t)$ of Table 5 were used.

The CHOA optimizer aims to produce a CNN network with the best performance so that the accuracy values of the test samples are acceptable. The CHOA optimizer runs 10 times, and the location of each cheetah is randomly determined in each run.

5.3.1. The first part of the experiments (attacks $f_1(y_t)$ to $f_6(y_t)$)

The CHOA optimizer is implemented with the ISSDA dataset. This dataset, after adding 6 abnormal samples, contains 6420 records, with 5136 records (80%) being training data and 1284 records (20) being test data.

Table 10 shows the accuracy values of each run. As it turns out, the accuracy value of the run 6 has the highest fitness value. Table 11 also shows the hyper-parameters obtained in 10 different runs.

Table 10. Fitness values of 10 runs

No Run / Fitness	1	2	3	4	5	6	7	8	9	10
Accuracy	0.9984	0.9976	0.9976	0.9968	0.9984	0.9992	0.9961	0.9984	0.9945	0.9976

Table 11. Hyper parameters of 10 different runs

No Of runs	No layer	Layer 1					Layer 2					Layer 3					Batch size
		No Filter	Filter size	Stride	Pool size	Stride	No filter	Filter size	stride	Pool size	stride	No Filter	Filter size	Stride	Pool size	Stride	
1	2	30	2	3	3	4	64	3	2	2	2	0	0	0	0	0	100
2	2	32	2	4	1	3	60	2	2	3	4	0	0	0	0	0	120
3	3	30	3	2	2	2	64	3	4	3	2	118	2	2	3	3	98
4	3	32	3	4	2	2	58	3	3	2	2	120	2	3	3	4	118
5	3	28	3	3	3	3	64	3	2	2	3	120	2	3	3	2	122
6	3	32	2	2	2	2	64	2	3	2	2	128	2	2	3	2	128
7	2	32	3	1	3	4	60	3	1	3	4	0	0	0	0	0	124
8	2	30	2	2	3	1	54	2	2	2	4	0	0	0	0	0	128
9	3	32	2	2	2	2	58	2	3	3	1	128	3	2	3	2	128
10	3	30	2	4	2	3	64	2	3	3	2	128	2	3	3	3	114

Fig 7 illustrates the convergence of the CHOA algorithm over 10 different runs. According to Fig 7, the proposed framework converges in the range between iteration 10 and iteration 20, which shows the high performance and speed of the proposed method in achieving the optimal result.

These runs are performed with 100 iterations. The final CNN network for attacks $f_1(y_t)$ to $f_6(y_t)$is shown with the highest accuracy in Fig 8.

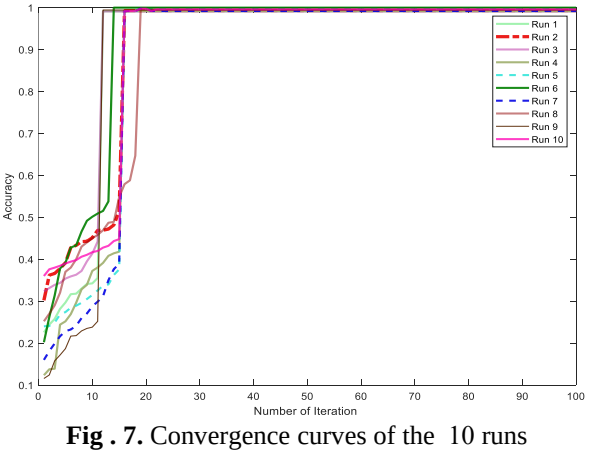

Fig . 7. Convergence curves of the 10 runs

In another experiment, the accuracy of the proposed framework was compared with SVM, CNN and PSO-CNN.

Table 12. Comparison of the results of the proposed framework with other methods

Method / Criterion	SVM	CNN	PSO-CNN	CHOA-CNN
Accuracy	0.8862	0.9517	0.9711	0.9992

Table 13. Performance comparison of proposed framework with/ without clustering

Method / Criterion	without clustering	With clustering
Accuracy	0.9665	0.9992

The results of this comparison were shown in Table 12. According to the findings, the proposed framework outperforms the other three methods in terms of accuracy.

As mentioned earlier, clustering of artificially abnormal data is effective in the end result of the proposed framework. Table 13 shows a comparison of the results of the proposed framework for two scenarios: utilizing the CHOA clustering algorithm or without using it. As shown in Table 13, the accuracy of the proposed framework was increased for situation where clustering was used. This accuracy was 0.9665 for situation where clustering was not used and 0.9992 for situation where the CHOA algorithm was used for clustering.

In the latest experiment of this part, the results of the proposed framewor were compared with other research studies conducted to detect electricity theft.

Table 14. Performance Comparison of proposed framework with other research studies

Criterion / Research	Accuracy	Recall	Precision	F-score
Proposed framework	0.9992	0.9992	0.9996	0.9994
Research [6]	0.996	0.9984	0.9969	0.9977
Research [2]	0.9933	0.981	0.972	0.977
Research [10]	0.98	0.9703	0.9703	0.9703
Research [15]	0.958	1	0.857	0.947

According to Table 14, the proposed framework with high accuracy identifies abnormal customers.

5.3.2. The second part of the experiments (attacks $f_1(y_t)$ to $f_9(y_t)$)

In this section, the performance of the proposed framework is evaluated using the 9 attacks, as presented in

Table 1. After adding 9 abnormal samples, contains 9630 records, with 7704 records (80%) being training data and 1926 records (20) being test data.

For evaluation, 10 independent runs were used. Table 15 shows the accuracy values of 10 different runs. According to Table 15, it can be seen that all runs have high accuracy and run8 have the highest accuracy. Table 16 also shows the extracted hyper-parameters of each run.

Table 15. Fitness values of 10 runs

No Run / Fitness	1	2	3	4	5	6	7	8	9	10
Accuracy	0.9974	0.9984	0.9958	0.9906	0.9974	0.9942	0.9968	0.9989	0.9984	0.9974

Table 16. Hyper parameters of 10 different runs

No Of runs	No layer	Layer 1					Layer 2					Layer 3					Batch size
		No filter	Filter size	Stride	Pool size	stride	No filter	Filter size	stride	Pool size	stride	No Filter	Filter size	Stride	Pool size	Stride	
1	3	18	2	2	2	2	64	2	2	3	2	64	2	2	3	2	124
2	2	26	2	2	2	2	56	2	3	2	3	0	0	0	0	0	130
3	3	32	2	3	2	2	32	2	2	2	3	96	2	2	2	3	100
4	2	30	2	2	2	2	30	2	2	3	2	0	0	0	0	0	94
5	3	30	2	3	2	2	28	2	3	2	3	116	2	3	2	2	120
6	2	16	3	2	3	2	58	2	2	2	2	0	0	0	0	0	48
7	3	24	2	2	2	2	64	2	3	2	2	84	2	3	2	2	92
8	3	32	2	3	2	2	64	2	3	2	2	118	2	2	2	2	108
9	3	16	2	2	2	3	38	2	2	3	2	76	2	2	3	2	98
10	3	18	3	3	2	3	56	2	2	3	2	122	2	3	2	2	120

Fig 9 illustrates the convergence process of 10 independent runs. It can be seen that all runs converge to the solution in the lowest possible iteration.

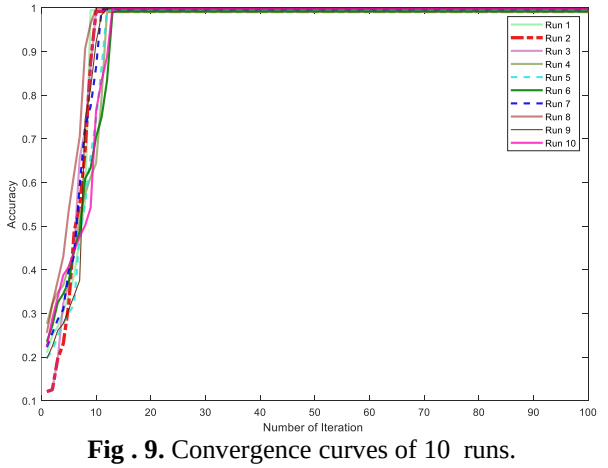

Fig . 9. Convergence curves of 10 runs.

The final CNN network for attacks $f_1(y_t)$ to $f_9(y_t)$ is shown with the highest accuracy in Fig 10. In the last experiment of this part, the performance of the proposed framework was compared with SVM, CNN and PSO-CNN methods. According to Table 17, the performance of the proposed framework is better than the other two methods

.

Table 17. Result comparisons of proposed framework with other methods

Method / Criterion	SVM	CNN	PSO-CNN	CHOA-CNN
Accuracy	0.8722	0.9714	0.9787	0.9989

According to the results of several experiments, it was observed that the proposed framework, with high accuracy, identifies electricity theft detection.

6. CONCLUSION

Due to the importance of early detection of electrical theft, CNN was used to improve the detection rate and increase accuracy in this study. Given that this type of network uses many hyper-parameters, manually setting these hyper-parameters is difficult and illogical. For automatic adjusting, CHOA algorithm was used in combination with CNN. In order to achieve better results and increase the correct detection rate, the dataset was balanced with artificial attacks, and then the CHOA algorithm was used to cluster and specify the abnormal class label. The accuracy of the proposed framework was evaluated by several experiments. The results showed that the proposed framework with high accuracy identified abnormal customers.

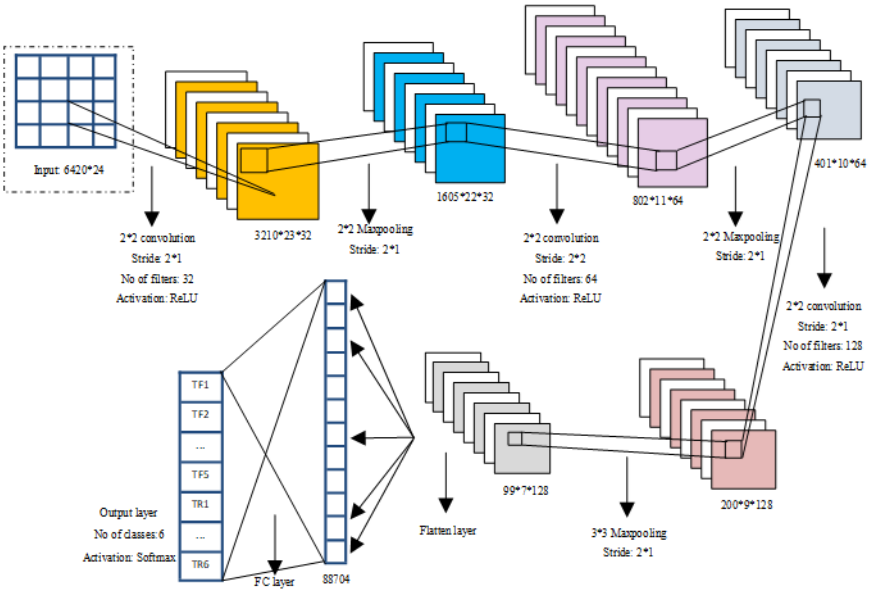

Fig. 8. Optimal CNN architecture produced using CHOA algorithm

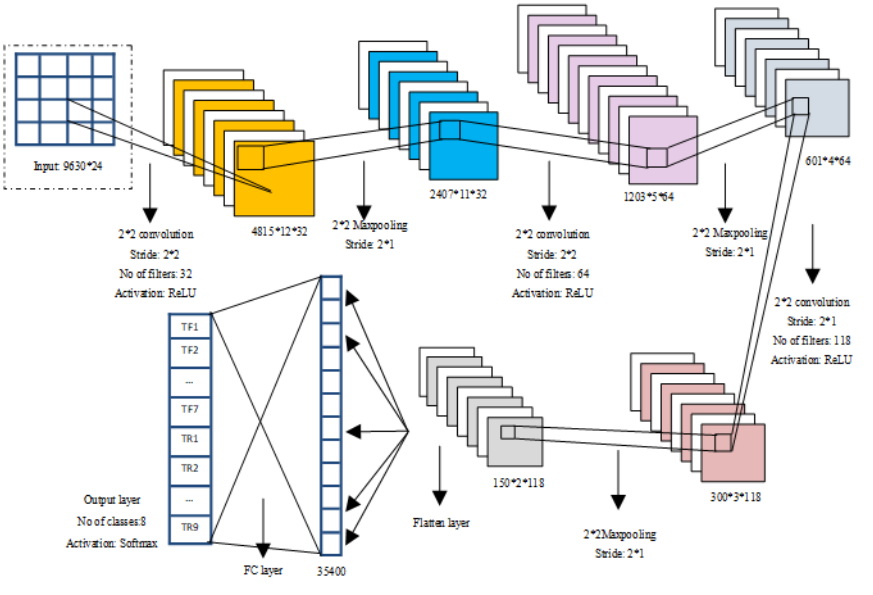

Fig. 10. Optimal CNN architecture produced using CHOA algorithm

REFERENCES

[1] P. Jokar, N. Arianpoo, and V. C. M. Leung, "**Electricity theft detection in AMI using customers' consumption patterns**," *IEEE Trans. Smart Grid*, Vol. 7, No. 1, pp. 216–226, 2016, doi: 10.1109/TSG.2015.2425222.

[2] H. Ghaedi, S. R. K. Tabbakh, and R. Ghaemi, "Improving Electricity Theft Detection using Combination of Improved Crow Search Algorithm and Support Vector Machine," *Majlesi J. Electr. Eng.*, vol. 15, no. 4 SE-Articles, Dec. 2021, doi: https://doi.org/10.52547/mjee.15.4.63.

[3] A. Askarzadeh, "**A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm**," *Comput. Struct.*, Vol. 169, pp. 1–12, 2016, doi:

10.1016/j.compstruc.2016.03.001.

[4] R. Razavi, A. Gharipour, M. Fleury, and I. J. Akpan, "**A practical feature-engineering framework for electricity theft detection in smart grids**," *Appl. Energy*, Vol. 238, No. December 2018, pp. 481–494, 2019, doi: 10.1016/j.apenergy.2019.01.076.

[5] Z. Feng, J. Huang, W. H. Tang, and M. Shahidehpour, "**Data mining for abnormal power consumption pattern detection based on local matrix reconstruction,"** *Int. J. Electr. Power Energy Syst.*, Vol. 123, No. February, p. 106315, 2020, doi: 10.1016/j.ijepes.2020.106315.

[6] H. Ghaedi, S. R. Kamel Tabbakh Farizani, and R. Gaemi, "**A Novel Meta-heuristic Framework for Solving Power Theft Detection Problem: Cheetah Optimization Algorithm,**" *Int. J. Ind. Electron. Control Optim.*, Vol. 5, No. 1, pp. 63–76, 2022, doi: 10.22111/ieco.2022.39528.1370.

[7] A. Ullah, N. Javaid, A. S. Yahaya, T. Sultana, F. A. Al-Zahrani, and F. Zaman, "**A Hybrid Deep Neural Network for Electricity Theft Detection Using Intelligent Antenna-Based Smart Meters,**" *Wirel. Commun. Mob. Comput.*, Vol. 2021, p. 9933111, 2021, doi: 10.1155/2021/9933111.

[8] N. M. Ibrahim, S. T. F. Al-Janabi, and B. Al-Khateeb, "**Electricity-theft detection in smart grids based on deep learning,**" *Bull. Electr. Eng. Informatics*, Vol. 10, No. 4, pp. 2285–2292, 2021, doi: 10.11591/EEI.V10I4.2875.

[9] M. Nazmul Hasan, R. N. Toma, A. Al Nahid, M. M. Manjurul Islam, and J. M. Kim, "E**lectricity theft detection in smart grid systems: A CNN-LSTM based approach,**" *Energies*, Vol. 12, No. 17, pp. 1–18, 2019, doi: 10.3390/en12173310.

[10] S. Li, Y. Han, X. Yao, S. Yingchen, J. Wang, and Q. Zhao, "**Electricity Theft Detection in Power Grids with Deep Learning and Random Forests,**" *J. Electr. Comput. Eng.*, Vol. 2019, 2019, doi: 10.1155/2019/4136874.

[11] B. Kocaman and V. Tümen, "**Detection of electricity theft using data processing and LSTM method in distribution systems,**" *Sadhana - Acad. Proc. Eng. Sci.*, Vol. 45, No. 1, 2020, doi: 10.1007/s12046-020-01512-0.

[12] H. M. Rouzbahani, H. Karimipour, and L. Lei, "**An Ensemble Deep Convolutional Neural Network Model for Electricity Theft Detection in Smart Grids,**" *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, Vol. 2020-Octob, pp. 3637–3642, 2020, doi: 10.1109/SMC42975.2020.9282837.

[13] Z. Zheng, Y. Yang, X. Niu, H. N. Dai, and Y. Zhou, "**Wide and Deep Convolutional Neural Networks for Electricity-Theft Detection to Secure Smart Grids**," *IEEE Trans. Ind. Informatics*, Vol. 14, No. 4, pp. 1606–1615, 2018, doi: 10.1109/TII.2017.2785963.

[14] P. Chandel and T. Thakur, "**Smart Meter Data Analysis for Electricity Theft Detection using Neural Networks,**" *Adv. Sci. Technol. Eng. Syst. J.*, Vol. 4, No. 4, pp. 161–168, 2019, doi: 10.25046/aj040420.

[15] X. Feng *et al.*, "**A novel electricity theft detection scheme based on text convolutional neural networks**," *Energies*, Vol. 13, No. 21, pp. 1–17, 2020, doi: 10.3390/en13215758.

[16] A. Maamar and K. Benahmed, "**A Hybrid Model for Anomalies Detection in AMI System Combining K-means Clustering and Deep Neural Network**," *Comput. Mater. Contin.*, Vol. 60, No. 1, pp. 15–39, 2019, doi: 10.32604/cmc.2019.06497.

[17] Z. Fouad, M. Alfonse, M. Roushdy, and A. B. M. Salem, "Hyper-parameter optimization of convolutional neural network based on particle swarm optimization algorithm," *Bull. Electr. Eng. Informatics*, Vol. 10, No. 6, pp. 3377–3384, 2021, doi: 10.11591/eei.v10i6.3257.

[18] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, and M. Tuba, "**Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics**," *Algorithms*, Vol. 13, No. 3, 2020, doi: 10.3390/a13030067.

[19] A. Brodzicki, M. Piekarski, and J. Jaworek-Korjakowska, "T**he whale optimization algorithm approach for deep neural networks**," *Sensors*, Vol. 21, No. 23, 2021, doi: 10.3390/s21238003.

[20] M. Han, A. Tan, and J. Zhong, "**Application of Particle Swarm Optimization Combined with Long and Short-term Memory Networks for Short-term Load Forecasting**," *J. Phys. Conf. Ser.*, Vol. 2203, No. 1, 2022, doi: 10.1088/1742-6596/2203/1/012047.

[21] S. H. Kim, Z. W. Geem, and G. T. Han, "**Hyperparameter optimization method based on harmony search algorithm to improve performance of 1D CNN human respiration pattern recognition system,**" *Sensors (Switzerland)*, Vol. 20, No. 13, pp. 1–20, 2020, doi: 10.3390/s20133697.

[22] T. Serizawa and H. Fujita, "**Optimization of Convolutional Neural Network Using the Linearly Decreasing Weight Particle Swarm Optimization,**" *ArXiv*, Vol. abs/2001.0, 2020, [Online]. Available: http://arxiv.org/abs/2001.05670.

[23] C. Li *et al.*, "**Genetic Algorithm based hyper-parameters optimization for transfer**

Convolutional Neural Network," *arXiv Prepr. arXiv2103.03875*, Vol. abs/2103.0, 2021.

[24] W.-C. Yeh, Y.-P. Lin, Y.-C. Liang, C.-M. Lai, and X.-Z. Gao, "**Simplified Swarm Optimisation for the Hyperparameters of a Convolutional Neural Network,**" pp. 1–26.

[25] M. Zhou, "**Heuristic Hyperparameter Optimization for Convolutional Neural Networks using Genetic Algorithm**," pp. 1–8, 2021, [Online]. Available: http://arxiv.org/abs/2112.07087.

[26] K. Simonyan and A. Zisserman, "**Very deep convolutional networks for large-scale image recognition**," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[27] [X. Kan *et al.*, "**A novel PSO-based optimized lightweight convolution neural network for movements recognizing from multichannel surface electromyogram,**" *Complexity*, vol. 2020, 2020, doi: 10.1155/2020/6642463.

[28] F. Johnson, A. Valderrama, C. Valle, B. Crawford, R. Soto, and R. Nanculef, "**Automating Configuration of Convolutional Neural Network Hyperparameters Using Genetic Algorithm**," *IEEE Access*, Vol. 8, pp. 156139–156152, 2020, doi: 10.1109/ACCESS.2020.3019245.

[29] N. M. Aszemi and P. D. D. Dominic, "**Hyperparameter optimization in convolutional neural network using genetic algorithms**," *Int. J. Adv. Comput. Sci. Appl.*, Vol. 10, No. 6, pp. 269–278, 2019, doi: 10.14569/ijacsa.2019.0100638.

[30] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "**Convolutional neural networks: an overview and application in radiology**," *Insights Imaging*, Vol. 9, No. 4, pp. 611–629, 2018, doi: 10.1007/s13244-018-0639-9.

[31] S. Khan, H. Rahmani, S. A. A. Shah, M. Bennamoun, G. Medioni, and S. Dickinson, *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool, 2018.

[32] J. Raitoharju, "**Chapter 3 - Convolutional neural networks,**" A. Iosifidis and A. B. T.-D. L. for R. P. and C. Tefas, Eds. Academic Press, 2022, pp. 35–69.

[33] W. H. Lopez Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "**Chapter 10 - Convolutional neural networks**," A. Mechelli and S. B. T.-M. L. Vieira, Eds. Academic Press, 2020, pp. 173–191.

[34] Y. Tian, "**Artificial Intelligence Image Recognition Method Based on Convolutional Neural Network Algorithm,**" *IEEE Access*, vol. 8, pp. 125731–125744, 2020, doi: 10.1109/ACCESS.2020.3006097.

[35] V. Kotu and B. Deshpande, "**Chapter 10 - Deep Learning,**" V. Kotu and B. B. T.-D. S. (Second E. Deshpande, Eds. Morgan Kaufmann, 2019, pp. 307–342.

[36] "**Irish Social Science Data Archive**," 2012. https://www.ucd.ie/issda/data/commissionforenergyregulationcer/.

[37] S. Das, A. Abraham, and A. Konar, "**Automatic clustering using an improved differential evolution algorithm**," *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, Vol. 38, No. 1, pp. 218–237, 2008, doi: 10.1109/TSMCA.2007.909595.

[38] D. L. Davies and D. W. Bouldin, "**A Cluster Separation Measure,**" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, No. 2, pp. 224–227, 1979, doi: 10.1109/TPAMI.1979.4766909.